Mahidol University *Wisdom of the Land*

Chapter 9

Image Compression

Compression Fundamentals

- Data compression
  - the process of reducing the amount of data required to represent a given quantity of information.

- Data and Information
  - Note that data and information are not the same.
  - Data are the means by which information is conveyed.
  - Various amounts of data may be used to represent the same amount of information.

- Data that either provide no relevant information or simply restate that which is already known

  → data redundancy

Compression Fundamentals

- **Data vs Information**
  - Information = Matter
  - Data = The means by which information is conveyed

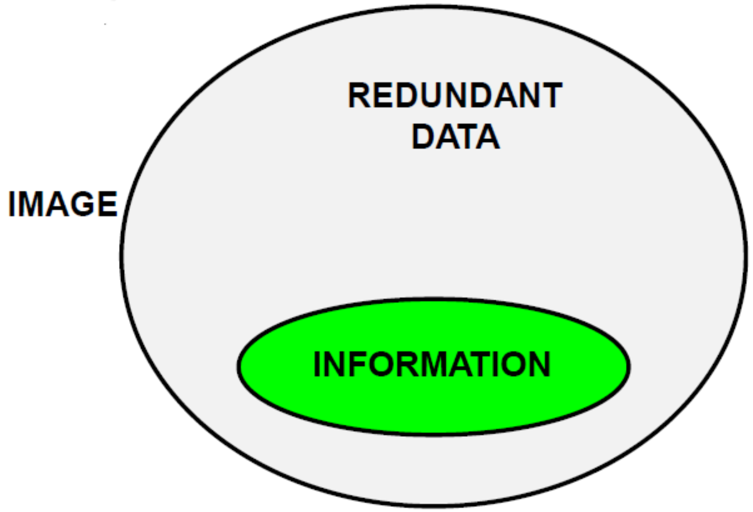
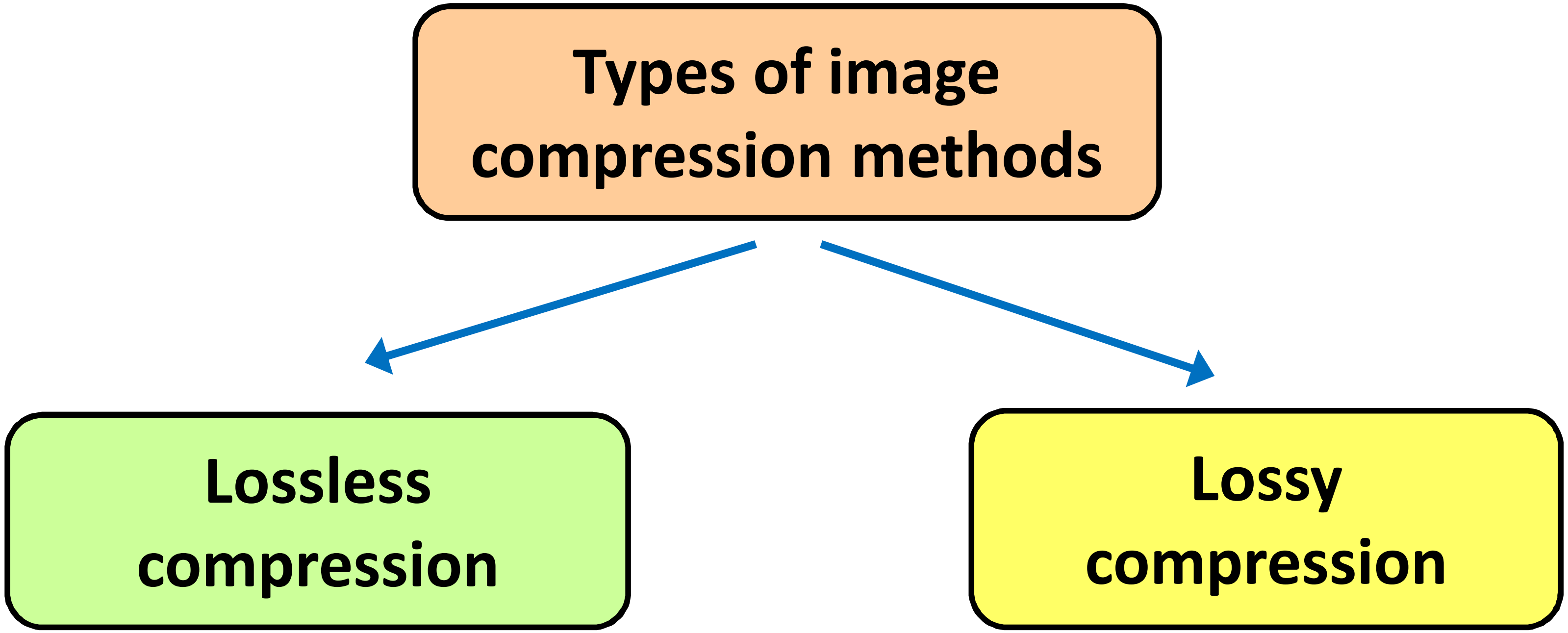


IMAGE = INFORMATION + REDUNDANT DATA

- **Image Compression**
  - Reducing the amount of data required to represent a digital image while keeping information as much as possible

Image Compression Methods



Types of image compression methods

Lossless compression

Lossy compression

Lossless Compression VS. Lossy Compression

Lossless compression	Lossy compression
- Information preserving - Low compression ratios - Zero data loss	- Not information preserving - High compression ratios - Some data loss
- Example: Fixed length coding and Variable length coding	- Example: JPEG compression, JPEG2000 compression.

Coding redundancy

- **Coding Redundancy** : A code is a system of symbols (i.e. bits) that represents information. Each piece of information is represented by a set of code symbols.
- The normalized histogram of a gray level image can be used in construction of codes to reduce the data used to represent it.

$$p_r(r_k) = \frac{n_k}{n} \qquad , k = 0,1,2,\ldots, L-1$$

where

r_k is the pixel values defined in the interval $[0,1]$

$p_r(r_k)$ is the probability of occurrence of r_k.

L is the number of gray levels.

n_k is the number of times that k^{th} gray level appears

n is the total number of pixels.

Coding redundancy

- Different coding methods yield different amount of data needed to represent the same information.
- Average number of bits required to represent each pixel is given by :

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

where

$l(r_k)$ is the number of bits used to represent each pixel of r_k.

Relative Data Redundancy and Compression Ratio

- Given n_1 and n_2 denoting the information-carrying units in two data sets that represent the same information/image.
- Relative data redundancy, R_D of the first data set, n_1 ,is defined by :

$$\boxed{R_D = 1 - \frac{1}{C_R}}$$

- C_R refers to the compression ratio and is defined by :

$$\boxed{C_R = \frac{n_1}{n_2}}$$

- If $n_1 = n_2$, then $C_R = 1$ and $R_D = 0$, indicating that the first representation of the information contains no redundant data.
- A typical compression ratio around 15 or(15:1) indicates that 85% of the data in the first data set is redundant.

Example : Fixed Length Coding

- Assuming there is an image 3 bit (*L*=8). There are gray levels distribution values as Table in which the gray levels range [0, *L*-1] = [0, 7].

r_k	$p_r(r_k)$	Fixed Length Coding	$l_r(r_k)$
$r_0 = 0$	0.19	000	3
$r_1 = 1/7$	0.25	001	3
$r_2 = 2/7$	0.21	010	3
$r_3 = 3/7$	0.16	011	3
$r_4 = 4/7$	0.08	100	3
$r_5 = 5/7$	0.06	101	3
$r_6 = 6/7$	0.03	110	3
$r_7 = 1$	0.02	111	3

Example : Fixed Length Coding

- The average number of bit used for fixed 3-bit code :

$$
\begin{aligned}
L_{avg} &= \sum_{k=0}^{L-1} l(r_k)p_r(r_k) \\
&= 3(0.19) + 3(0.25) + 3(0.21) + 3(0.16) + 3(0.08) + 3(0.06) + 3(0.03) + 3(0.02) \\
&= 3.1 \\
&= 3 \quad bits
\end{aligned}
$$

Example : Variable Length Coding

- Assuming there is an image 3 bit (L=8). There are gray levels distribution values as Table in which the gray levels range [0, L-1] = [0, 7].

r_k	$p_r(r_k)$	Variable Length Coding	$l_r(r_k)$
$r_0 = 0$	0.19	11	2
$r_1 = 1/7$	0.25	01	2
$r_2 = 2/7$	0.21	10	2
$r_3 = 3/7$	0.16	001	3
$r_4 = 4/7$	0.08	0001	4
$r_5 = 5/7$	0.06	00001	5
$r_6 = 6/7$	0.03	000001	6
$r_7 = 1$	0.02	000000	6

Example : Variable Length Coding

- The average number of bit used for variable length code :

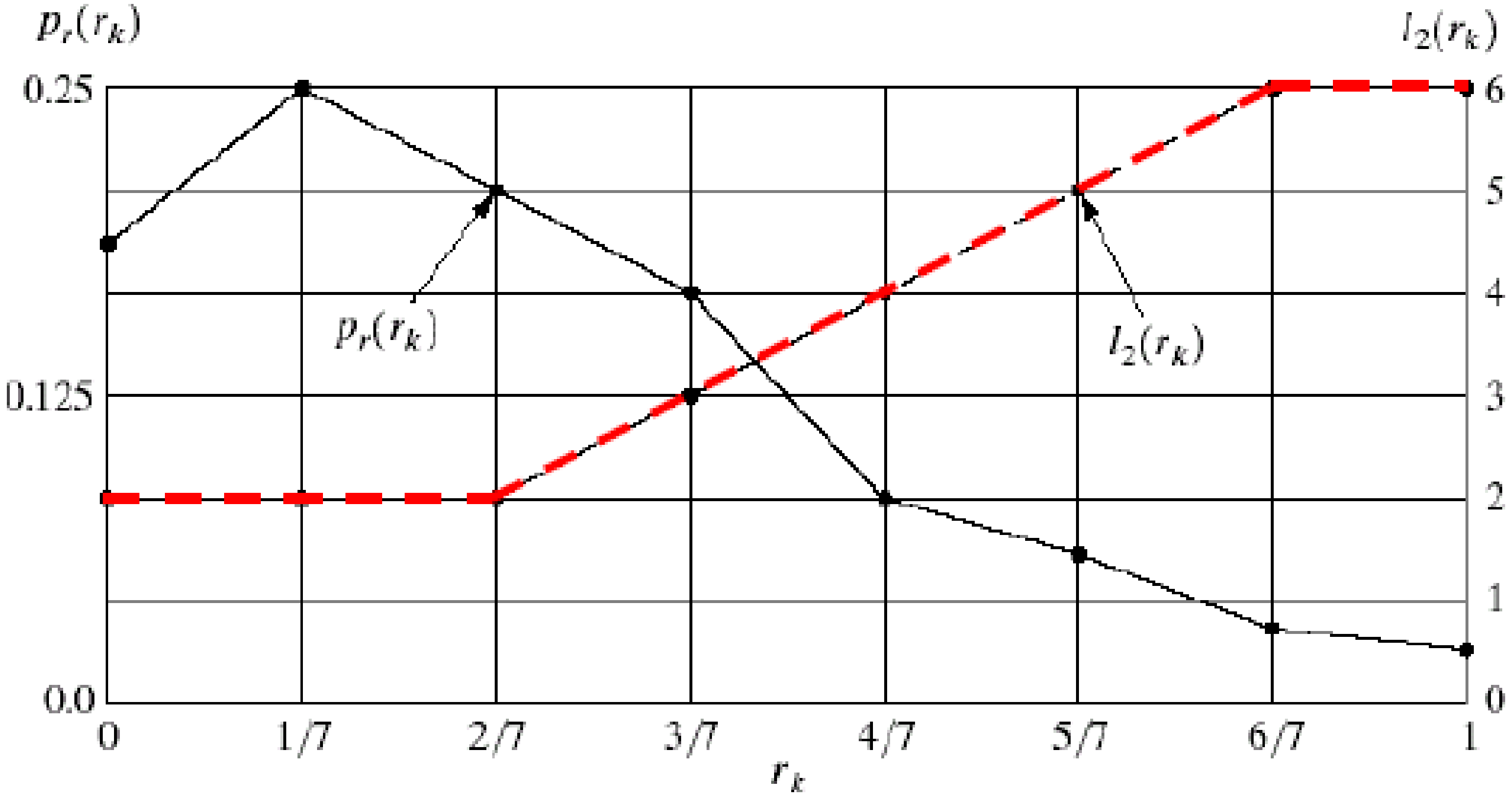
$$L_{avg} = \sum_{k=0}^{L-1} l(r_k)p_r(r_k)$$

$$= 2(0.19) + 2(0.25) + 2(0.21) + 3(0.16) + 4(0.08) + 5(0.06) + 6(0.03) + 6(0.02)$$

$$= 2.7 \quad bits$$

Example : Variable Length Coding

- Data compression is achieved by assigning fewer bits to more probable gray levels than the less probable gray levels.



Concept : assign the longest code word to the symbol with the least probability of occurrence.

Fixed Length Coding VS. Variable Length Coding

- Assuming there is an image 3 bit (L=8). There are gray levels distribution values as Table in which the gray levels range [0, L-1] = [0, 7].

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

Fixed 3-bit code

Variable length code

Fixed Length Coding VS. Variable Length Coding

- The average number of bit used for fixed 3-bit code :

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

$$= 3(0.19) + 3(0.25) + 3(0.21) + 3(0.16) + 3(0.08) + 3(0.06) + 3(0.03) + 3(0.02)$$

$$= 3 \quad bits$$

- The average number of bit used for variable length code :

$$L_{avg} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

$$= 2(0.19) + 2(0.25) + 2(0.21) + 3(0.16) + 4(0.08) + 5(0.06) + 6(0.03) + 6(0.02)$$

$$= 2.7 \quad bits$$

Fixed Length Coding VS. Variable Length Coding

- The compression ratio :

$$C_R = \frac{n_1}{n_2}$$

Fixed 3-bit code

Variable length code

$$= \frac{3}{2.7} = 1.11$$

- The relative Data Redundancy :

$$R_D = 1 - \frac{1}{C_R}$$

$$= 1 - \frac{1}{1.11} = 0.099 \Rightarrow \quad \cong \%10$$

Elements of Information Theory

- **Measuring Information** : The information in an image can be modeled as a probabilistic process, where we first develop a statistical model of the image generation process. The information content (entropy) can be estimated based on this model.
- The information per source (symbol or pixel), which is also referred as entropy is calculated by :

$$e = -\sum_{j=1}^{J} P(a_j)\log_2 P(a_j)$$

- Where $P(a_j)$ refers to the source symbol/pixel probabilities. J refers to the number of symbols or different pixel values.

Measuring Information

- For example, given the following gray scale image :

16	16	16	90	164	238	238	238
16	16	16	90	164	238	238	238
16	16	16	90	164	238	238	238
16	16	16	90	164	238	238	238

Measuring Information

- The entropy of the given gray scale mage can be calculated by :

Gray Level	Count	Probability
16	12	3/8
90	4	1/8
161	4	1/8
238	12	3/8

- The entropy of this image is calculated by :

$$e = -\sum_{j=1}^{J} P(a_j)\log_2 P(a_j)$$

$$= -\left[(3/8)\log_2(3/8) + (1/8)\log_2(1/8) + (1/8)\log_2(1/8) + (3/8)\log_2(3/8)\right]$$

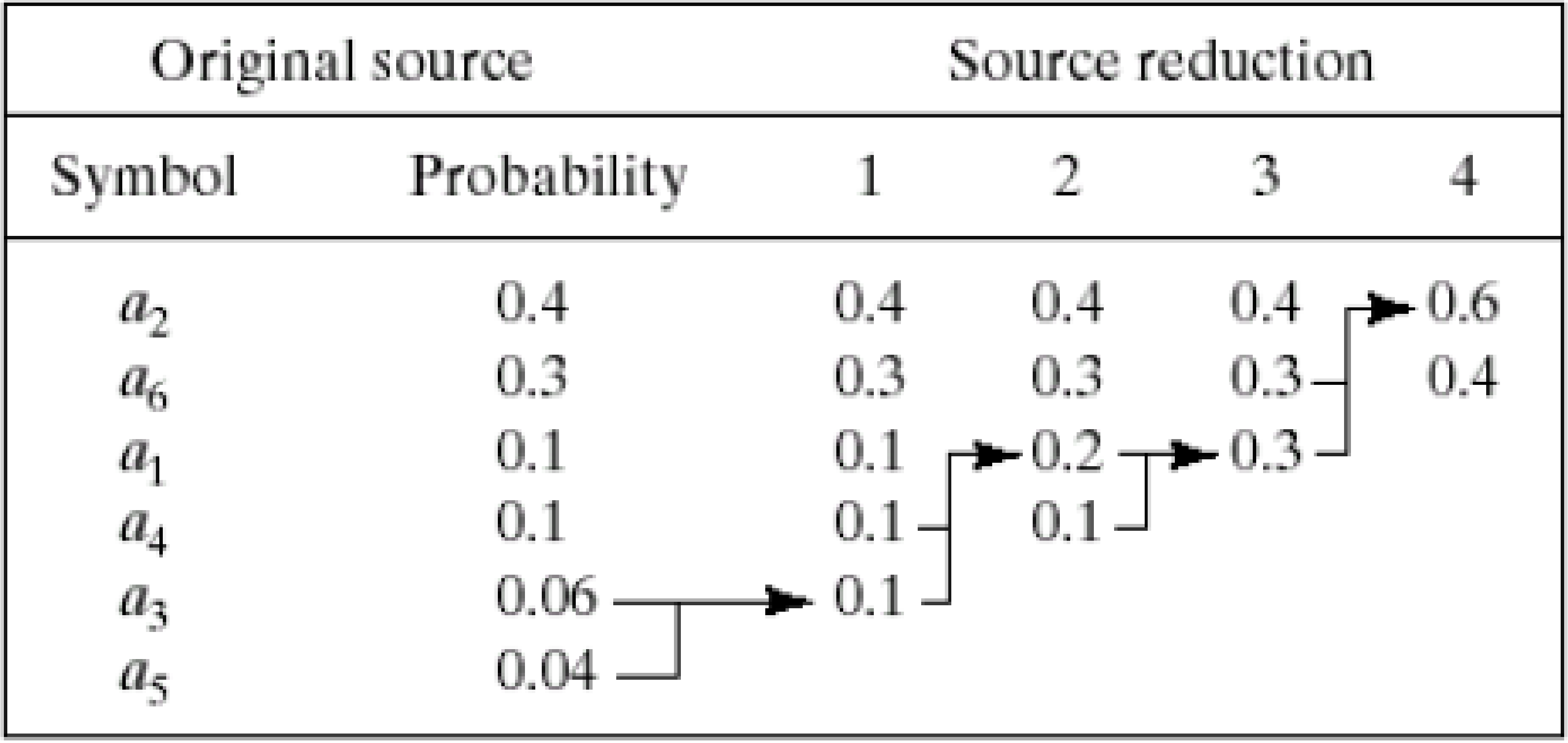
$$= 1.81 \quad bits/pixel$$

Huffman Coding

- **Huffman Coding** : The Huffman coding involves the following 2 steps.
  - 1) Create a series of source reductions by ordering the probabilities of the symbols and combining the lowest probability symbols into a single symbol and replace in the next source reduction.
  - 2) Code each reduced source starting with the smallest source and working back to the original source. Use 0 and 1 to code the simplest 2 symbol source.

Huffman Coding

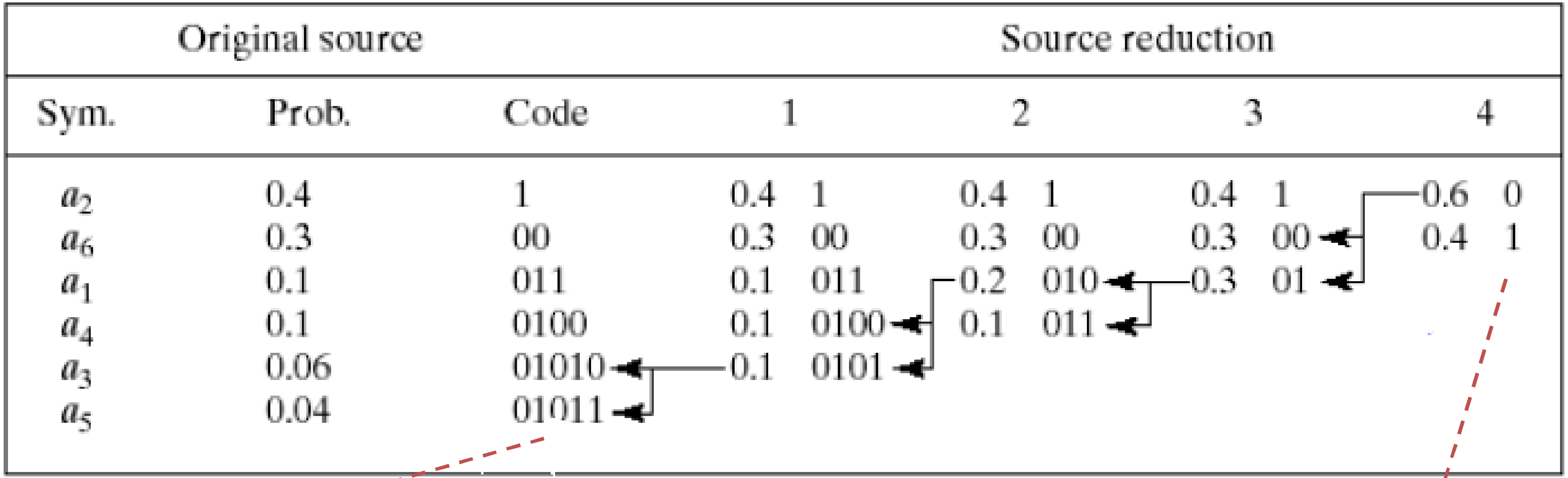
- 1) Huffman source reductions : a_i's corresponds to the available gray levels in a given image.



Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

Huffman Coding

- 2) Huffman code assignments : The first code assignment is done for a_2 with the highest probability and the last assignments are done for a_3 and a_5 with the lowest probabilities.



Original source			Source reduction							
Sym.	Prob.	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

Last code

First code

The code is instantaneous uniquely decodable without referencing succeeding symbols.

Huffman Coding

- Note that the shortest codeword (1) is given for the symbol/pixel with the highest probability (a_2). The longest codeword (01011) is given for the symbol/pixel with the lowest probability (a_2).
- The average length of the Huffman code is given by :

$$L_{avg} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5)$$

$$= 2.2 \quad bits / symbol$$

- The entropy of the Huffman code is given by :

$$e = -\sum_{j=1}^{J} P(a_j) \log_2 P(a_j)$$

$$= 2.14 \quad bits / symbol$$

Huffman Coding

- Huffman code efficiency is given by :

$$C = \frac{e}{L_{avg}} \times 100$$

where

L_{avg} is the average length of the Huffman code

e is the entropy of the Huffman code

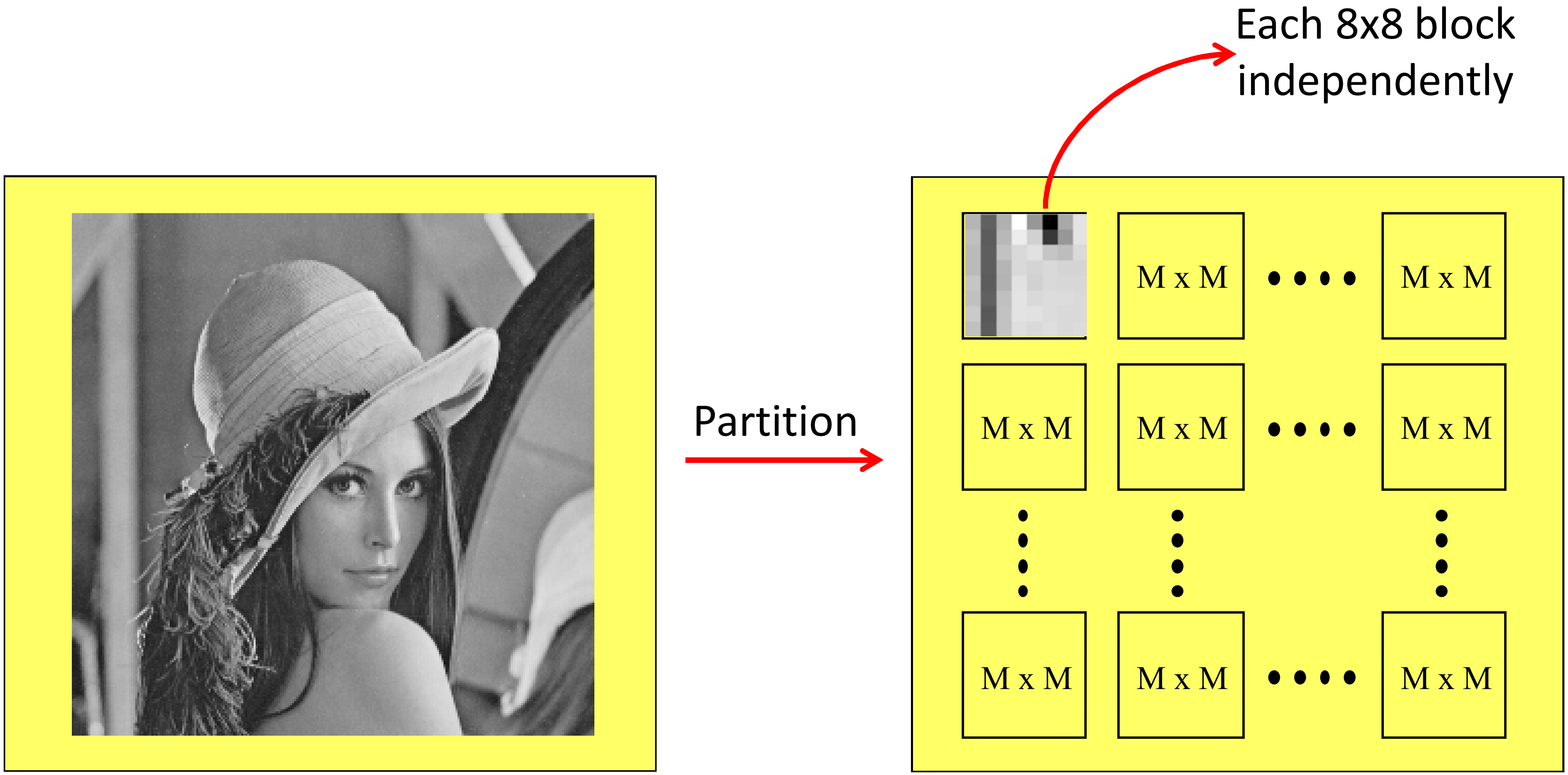
- The resulting Huffman coding efficiency is 97.3% ((2.14/2.2)*100).

JPEG compression

- Story of JPEG
  - Joint Photographic Expert Group
- International Organization for Standards (ISO)
  - 1988: ISO got together a group of experts to develop a good image compression scheme
    - Geared towards photographs of natural imagery
    - Color and monochrome
    - Easy to use (spin-dial quality control)
  - Through empirical testing, the following scheme proved to be the best
  - Standardized in August 1990

Block partition

- First, the image is divided into 8×8 non-overlapping blocks, which are processed left to right, top to bottom.



Each 8x8 block independently

Partition

M x M •••• M x M

M x M M x M •••• M x M

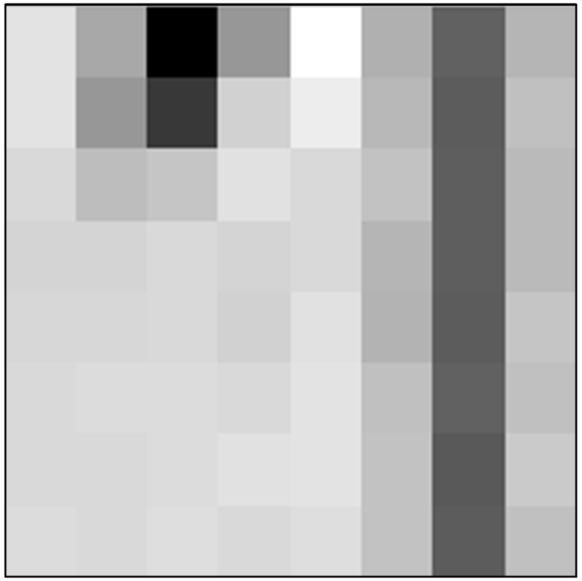
M x M M x M •••• M x M

Image

Image blocks

Example of image block

- As an example, 8×8 block of 8-bit image might be:



$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix}$$

Image block

Shift value procedure

- Before computing the DCT of the image block, its gray values are shifted from a positive range to one centered around zero.
- For an 8-bit image each pixel has 256 possible values: [0,255]. To center around zero it is necessary to subtract by half the number of possible values, or 128.

$$\begin{bmatrix} 183 & 160 & 94 & 153 & 194 & 163 & 132 & 165 \\ 183 & 153 & 116 & 176 & 187 & 166 & 130 & 169 \\ 179 & 168 & 171 & 182 & 179 & 170 & 131 & 167 \\ 177 & 177 & 179 & 177 & 179 & 165 & 131 & 167 \\ 178 & 178 & 179 & 176 & 182 & 164 & 130 & 171 \\ 179 & 180 & 180 & 179 & 183 & 169 & 132 & 169 \\ 179 & 179 & 180 & 182 & 183 & 170 & 129 & 173 \\ 180 & 179 & 181 & 179 & 181 & 170 & 130 & 169 \end{bmatrix} \rightarrow \boxed{-128} \rightarrow \begin{bmatrix} 55 & 32 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 41 & 4 & 41 \end{bmatrix}$$

- As an example, the first number is derived from 183 - 128 = 55.

Discrete Cosine Transform (DCT)

- 2D-DCT: $$G_{u,v} = \sum_{x=0}^{7}\sum_{y=0}^{7}\alpha(u)\alpha(v)g_{x,y}\cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$$

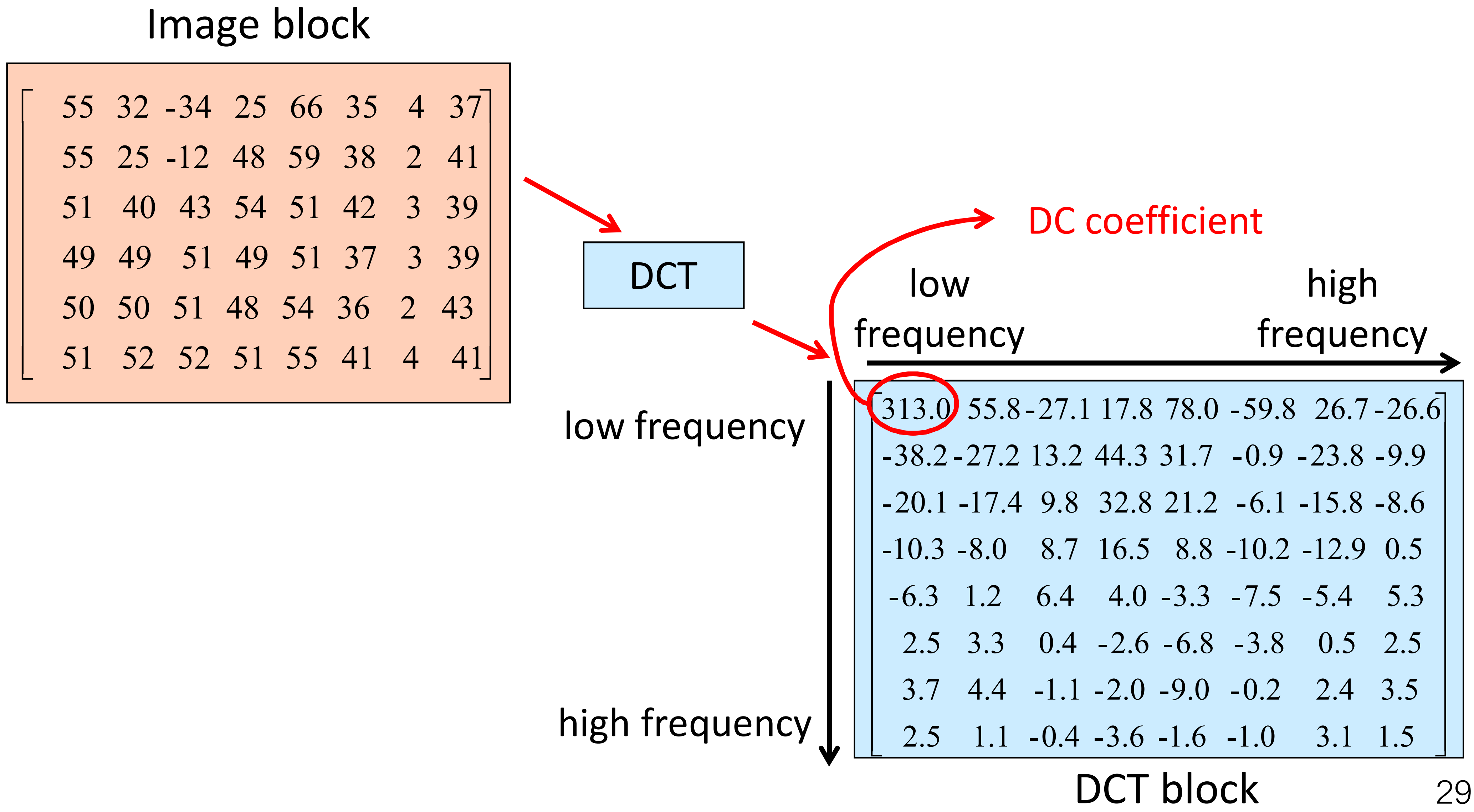


Image block

$$\begin{bmatrix} 55 & 32 & -34 & 25 & 66 & 35 & 4 & 37 \\ 55 & 25 & -12 & 48 & 59 & 38 & 2 & 41 \\ 51 & 40 & 43 & 54 & 51 & 42 & 3 & 39 \\ 49 & 49 & 51 & 49 & 51 & 37 & 3 & 39 \\ 50 & 50 & 51 & 48 & 54 & 36 & 2 & 43 \\ 51 & 52 & 52 & 51 & 55 & 41 & 4 & 41 \end{bmatrix}$$

$$\begin{bmatrix} 313.0 & 55.8 & -27.1 & 17.8 & 78.0 & -59.8 & 26.7 & -26.6 \\ -38.2 & -27.2 & 13.2 & 44.3 & 31.7 & -0.9 & -23.8 & -9.9 \\ -20.1 & -17.4 & 9.8 & 32.8 & 21.2 & -6.1 & -15.8 & -8.6 \\ -10.3 & -8.0 & 8.7 & 16.5 & 8.8 & -10.2 & -12.9 & 0.5 \\ -6.3 & 1.2 & 6.4 & 4.0 & -3.3 & -7.5 & -5.4 & 5.3 \\ 2.5 & 3.3 & 0.4 & -2.6 & -6.8 & -3.8 & 0.5 & 2.5 \\ 3.7 & 4.4 & -1.1 & -2.0 & -9.0 & -0.2 & 2.4 & 3.5 \\ 2.5 & 1.1 & -0.4 & -3.6 & -1.6 & -1.0 & 3.1 & 1.5 \end{bmatrix}$$

DCT block

Quantization table

- The human eye is good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation. This allows one to greatly reduce the amount of information in the high frequency components.
- The JPEG recommended luminance and chrominance quantization tables can be scaled to provide a variety of compression levels (select the quality of JPEG compression).

Luminance

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Chrominance

$$\begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

Quantization procedure

DCT block

$$\begin{bmatrix} 313.0 & 55.8 & -27.1 & 17.8 & 78.0 & -59.8 & 26.7 & -26.6 \\ -38.2 & -27.2 & 13.2 & 44.3 & 31.7 & -0.9 & -23.8 & -9.9 \\ -20.1 & -17.4 & 9.8 & 32.8 & 21.2 & -6.1 & -15.8 & -8.6 \\ -10.3 & -8.0 & 8.7 & 16.5 & 8.8 & -10.2 & -12.9 & 0.5 \\ -6.3 & 1.2 & 6.4 & 4.0 & -3.3 & -7.5 & -5.4 & 5.3 \\ 2.5 & 3.3 & 0.4 & -2.6 & -6.8 & -3.8 & 0.5 & 2.5 \\ 3.7 & 4.4 & -1.1 & -2.0 & -9.0 & -0.2 & 2.4 & 3.5 \\ 2.5 & 1.1 & -0.4 & -3.6 & -1.6 & -1.0 & 3.1 & 1.5 \end{bmatrix}$$

→ ÷ →

Quantization table

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

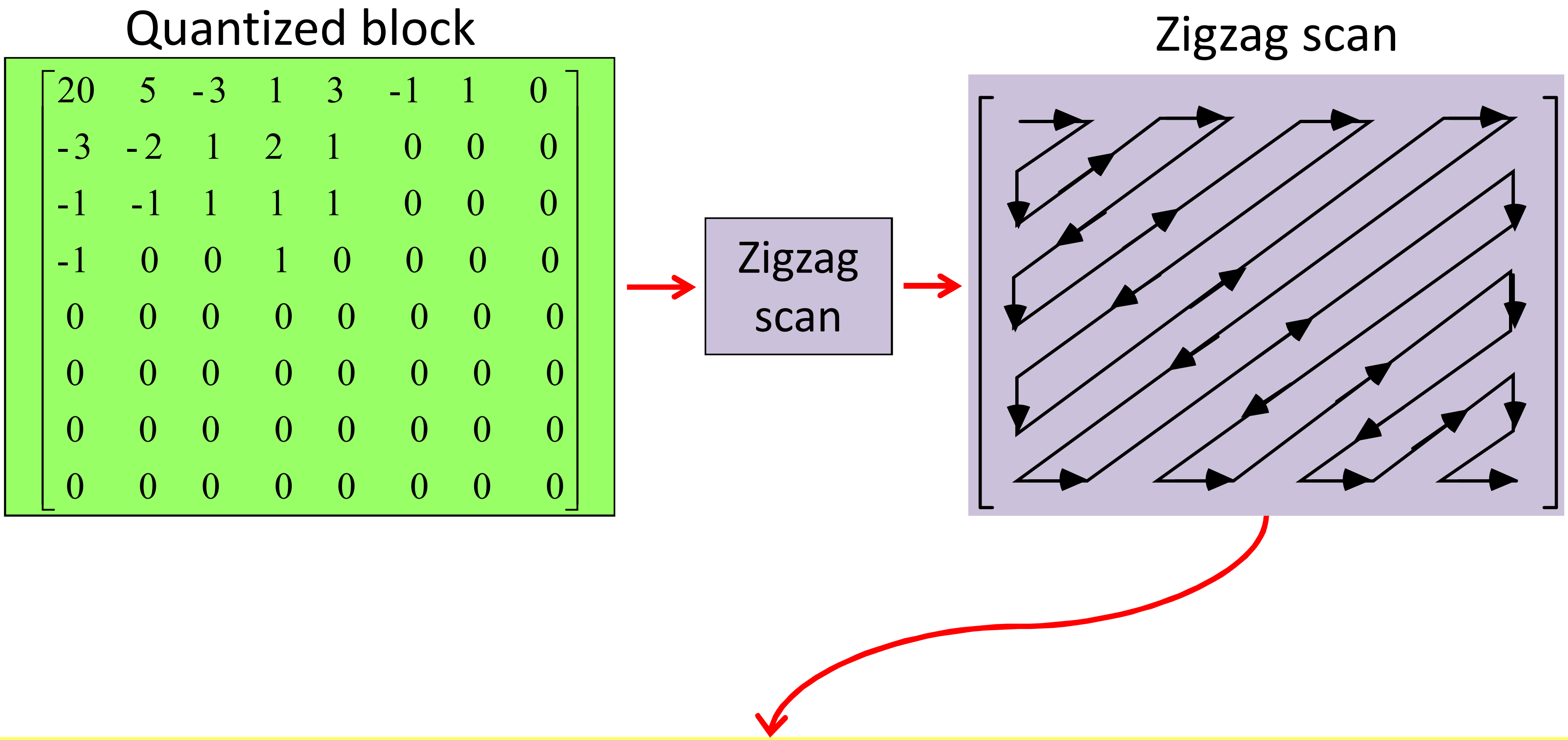
Round

Quantized block

$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -1 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\text{round}\left(\frac{313}{16}\right) = 20$$

Zigzag procedure



Quantized block

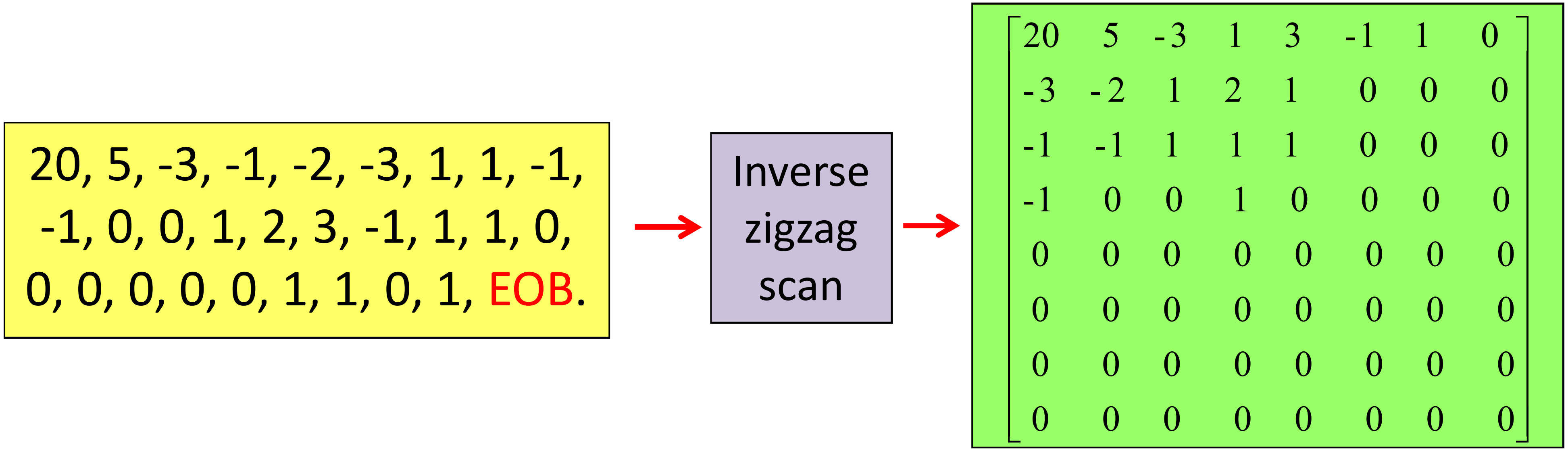
$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -1 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Zigzag scan

- Result = 20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 1, 2, 3, -1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, EOB.
- EOB symbol denotes the end-of-block condition.

Inverse zigzag procedure

- Decoding to display the image consists of doing all the above in reverse. Taking the DCT coefficient matrix (after adding the difference of the DC coefficient back in).



20, 5, -3, -1, -2, -3, 1, 1, -1, -1, 0, 0, 1, 2, 3, -1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, EOB.

Inverse zigzag scan

$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -1 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Inverse Quantization procedure

$$\begin{bmatrix} 20 & 5 & -3 & 1 & 3 & -1 & 1 & 0 \\ -3 & -2 & 1 & 2 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \rightarrow \times \rightarrow \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$\begin{bmatrix} 320 & 55 & -30 & 16 & 72 & -40 & 51 & 0 \\ -36 & -24 & 14 & 38 & 26 & 0 & 0 & 0 \\ -14 & -13 & 16 & 24 & 40 & 0 & 0 & 0 \\ -14 & 0 & 0 & 29 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Inverse Discrete Cosine Transform (IDCT)

- Inverse 2D-DCT: $f_{x,y} = \sum_{u=0}^{7}\sum_{v=0}^{7} \alpha(u)\alpha(v)F_{u,v} \cos\left[\frac{\pi}{8}\left(x+\frac{1}{2}\right)u\right]\cos\left[\frac{\pi}{8}\left(y+\frac{1}{2}\right)v\right]$

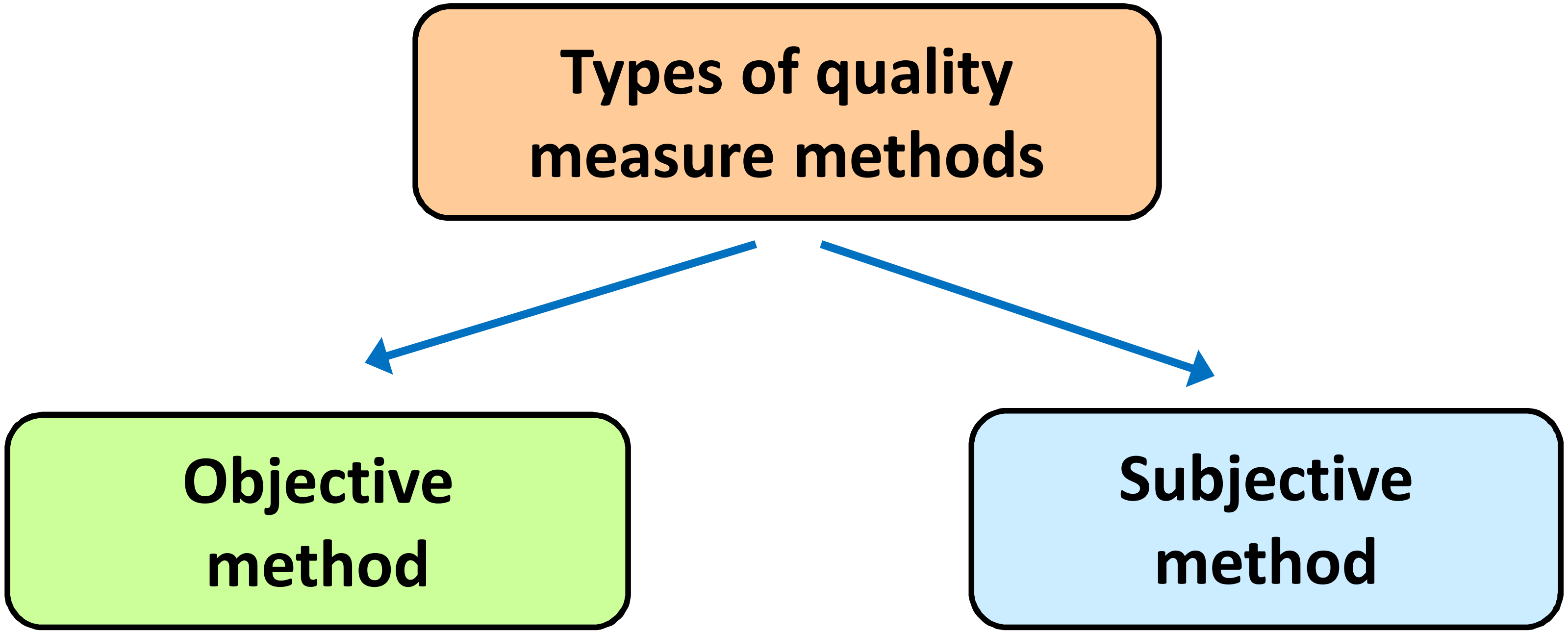
$$\begin{bmatrix}
320 & 55 & -30 & 16 & 72 & -40 & 51 & 0 \\
-36 & -24 & 14 & 38 & 26 & 0 & 0 & 0 \\
-14 & -13 & 16 & 24 & 40 & 0 & 0 & 0 \\
-14 & 0 & 0 & 29 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\xrightarrow{\text{IDCT}}
\begin{bmatrix}
71.2 & 4.7 & -7.2 & 25.4 & 63.2 & 41.2 & -1.3 & 37.9 \\
61.5 & 17.6 & 16.5 & 35.5 & 58.8 & 38.2 & 2.5 & 43.2 \\
50.1 & 34.3 & 45.3 & 46.1 & 53.4 & 37.1 & 7.3 & 45.1 \\
44.9 & 45.0 & 60.4 & 49.1 & 50.9 & 40.7 & 10.0 & 38.2 \\
47.5 & 47.1 & 59.5 & 46.3 & 52.1 & 45.7 & 9.9 & 28.9 \\
53.2 & 44.9 & 54.2 & 46.1 & 54.9 & 45.7 & 8.3 & 29.1 \\
57.2 & 43.5 & 54.3 & 51.9 & 56.9 & 39.4 & 6.7 & 40.9 \\
58.5 & 43.5 & 57.7 & 58.7 & 57.8 & 32.8 & 5.9 & 53.2
\end{bmatrix}$$

Inverse shift value procedure

- After that, rounding the output to integer values, and adding 128 to each entry. The decompression process may produce values outside of the original input range of [0,255].
- If this occurs, the decoder needs to clip the output values keep them within that range to prevent overflow when storing the decompressed image with the original bit depth.

$$\begin{bmatrix} 71.2 & 4.7 & -7.2 & 25.4 & 63.2 & 41.2 & -1.3 & 37.9 \\ 61.5 & 17.6 & 16.5 & 35.5 & 58.8 & 38.2 & 2.5 & 43.2 \\ 50.1 & 34.3 & 45.3 & 46.1 & 53.4 & 37.1 & 7.3 & 45.1 \\ 44.9 & 45.0 & 60.4 & 49.1 & 50.9 & 40.7 & 10.0 & 38.2 \\ 47.5 & 47.1 & 59.5 & 46.3 & 52.1 & 45.7 & 9.9 & 28.9 \\ 53.2 & 44.9 & 54.2 & 46.1 & 54.9 & 45.7 & 8.3 & 29.1 \\ 57.2 & 43.5 & 54.3 & 51.9 & 56.9 & 39.4 & 6.7 & 40.9 \\ 58.5 & 43.5 & 57.7 & 58.7 & 57.8 & 32.8 & 5.9 & 53.2 \end{bmatrix} \xrightarrow{\textbf{Round}} \boxed{+128} \rightarrow \begin{bmatrix} 199 & 133 & 121 & 153 & 191 & 169 & 127 & 166 \\ 190 & 146 & 145 & 163 & 187 & 166 & 130 & 171 \\ 178 & 162 & 173 & 174 & 181 & 165 & 135 & 173 \\ 173 & 173 & 188 & 177 & 179 & 169 & 138 & 166 \\ 175 & 175 & 187 & 174 & 180 & 174 & 138 & 157 \\ 181 & 173 & 182 & 174 & 183 & 174 & 136 & 157 \\ 185 & 171 & 182 & 180 & 185 & 167 & 135 & 169 \\ 186 & 172 & 186 & 187 & 186 & 161 & 134 & 181 \end{bmatrix}$$

Quality Measure of a Compressed Image



Types of quality measure methods

Objective method

Subjective method

Quality Measure of a Compressed Image

Objective quality measures (by formulate a criterion) :

- The mean-square error between an original image $\hat{f}(x,y)$ and compressed image $\hat{f}(x,y)$:

$$e_{MSE} = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2$$

where the images are of size *M* × *N*.

- Root-mean-square error :

$$e_{RMSE} = \sqrt{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2}$$

The smaller the value of e_{RMSE} , the better the compressed image represents the original image.

Quality Measure of a Compressed Image

- Mean-square signal-to-noise ratio :

$$SNR_{ms} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y)-f(x,y)\right]^2}$$

- Peak signal-to-noise ratio (*PSNR*) – in decibel (dB) :

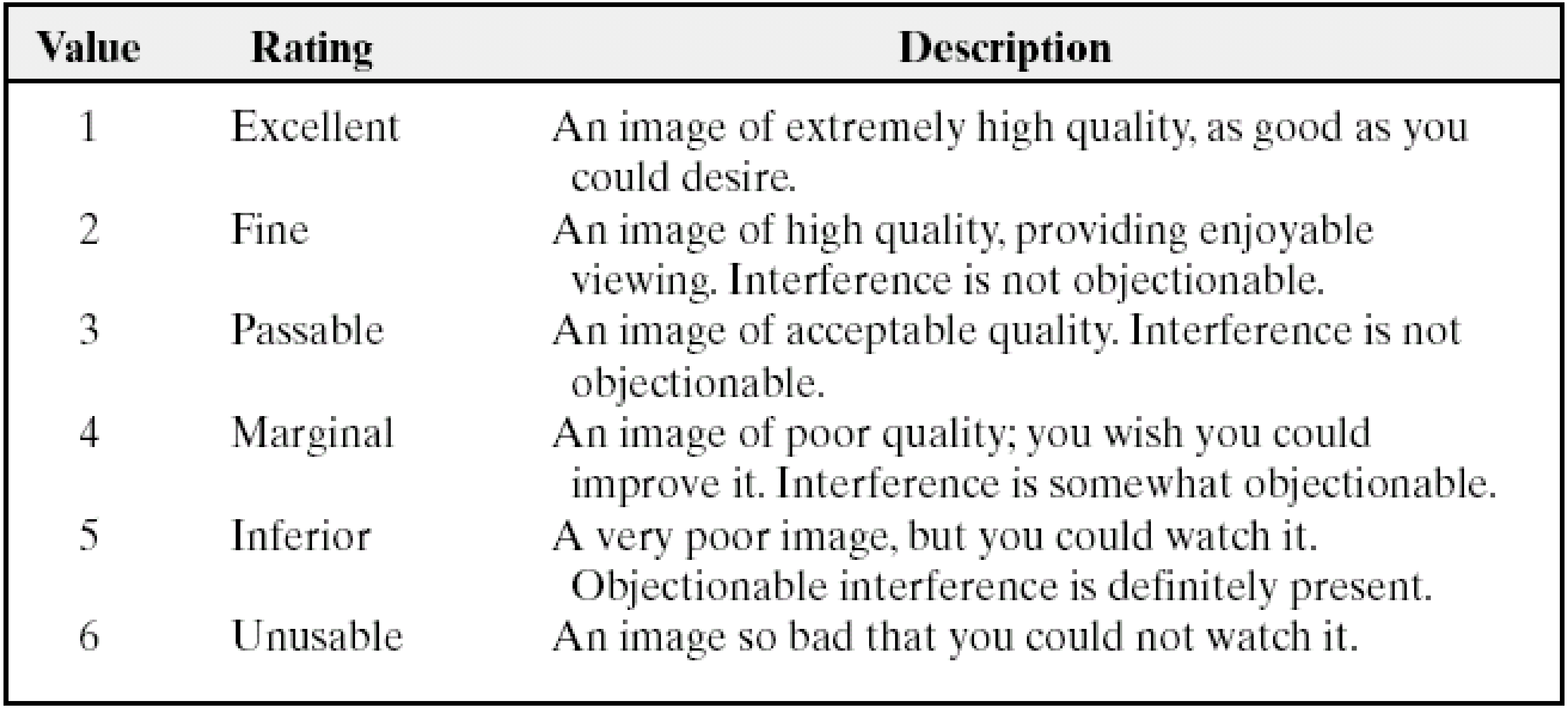
$$PSNR = 10(\log_{10})\frac{(L-1)^2}{e_{MSE}}$$

where *L* is the number of gray levels.

Quality Measure of a Compressed Image

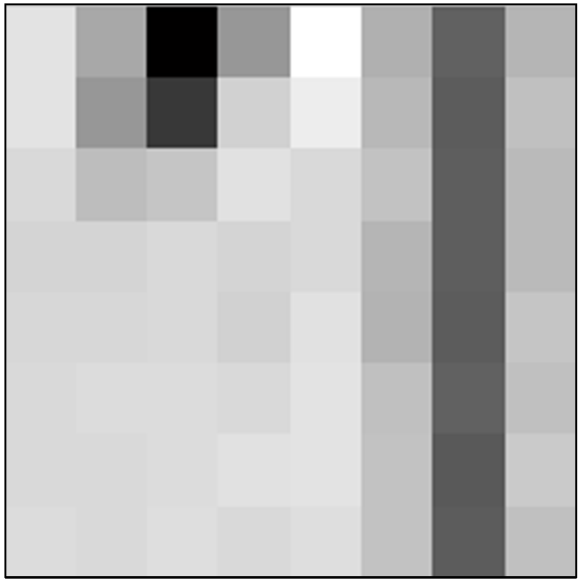
Subjective quality measures (by human observer):

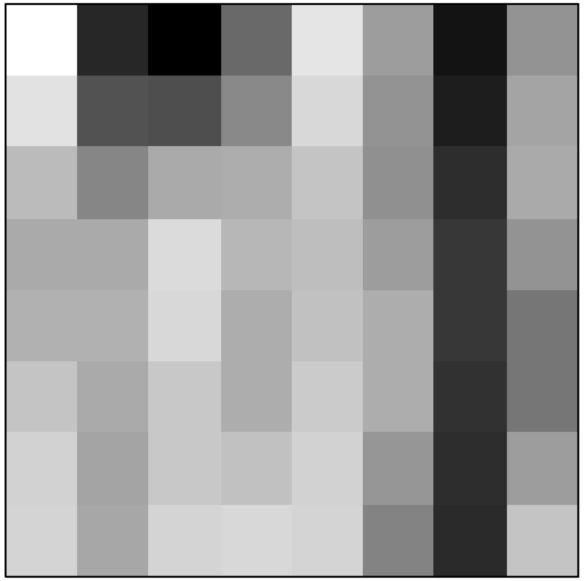
- This can be done by showing the compressed image to a group of viewers and averaging their evaluations. The evaluations may be made using an absolute rating scale.



Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

Original block VS. Decompressed block

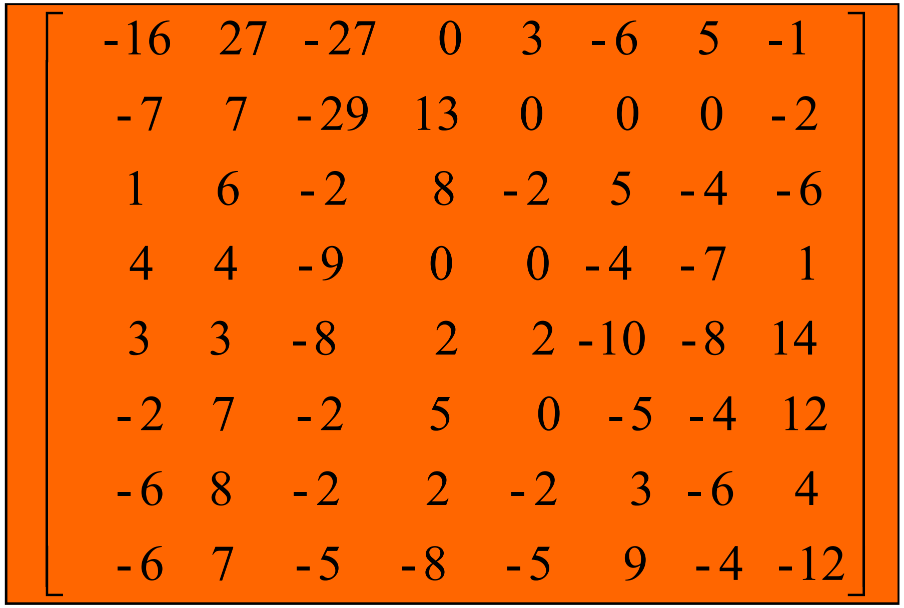




- Notice the slight differences between the original (left) and decompressed image (right), which is most readily seen in the bottom-left corner.

Difference results

- The decompressed block can be compared to the original block by taking the difference results in the following error values:



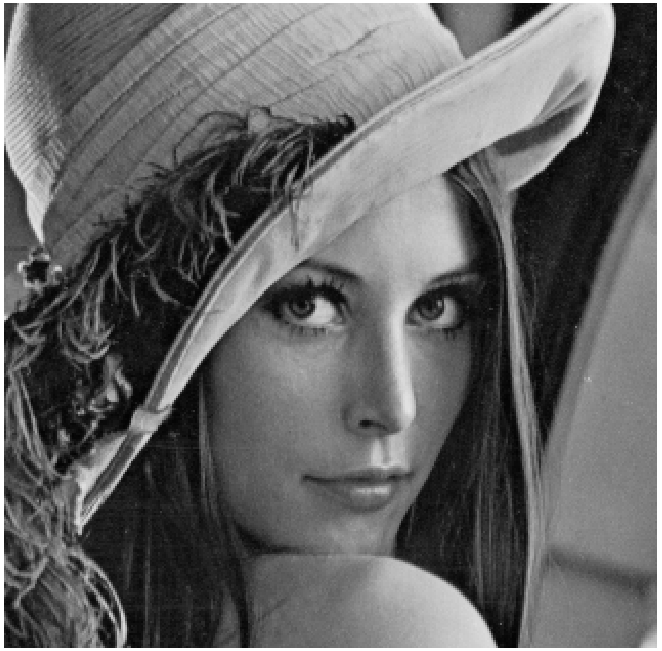
$$\begin{bmatrix} -16 & 27 & -27 & 0 & 3 & -6 & 5 & -1 \\ -7 & 7 & -29 & 13 & 0 & 0 & 0 & -2 \\ 1 & 6 & -2 & 8 & -2 & 5 & -4 & -6 \\ 4 & 4 & -9 & 0 & 0 & -4 & -7 & 1 \\ 3 & 3 & -8 & 2 & 2 & -10 & -8 & 14 \\ -2 & 7 & -2 & 5 & 0 & -5 & -4 & 12 \\ -6 & 8 & -2 & 2 & -2 & 3 & -6 & 4 \\ -6 & 7 & -5 & -8 & -5 & 9 & -4 & -12 \end{bmatrix}$$

- With an average absolute error of about 9 values per pixels.

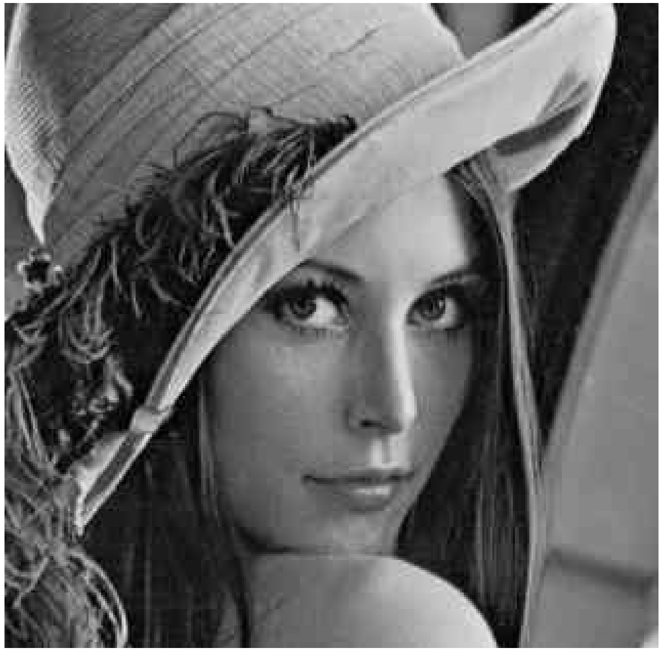
(i.e., $\frac{1}{64}\sum_{x=0}^{7}\sum_{y=0}^{7}|e(x,y)| = 8.4705$

JPEG compression results

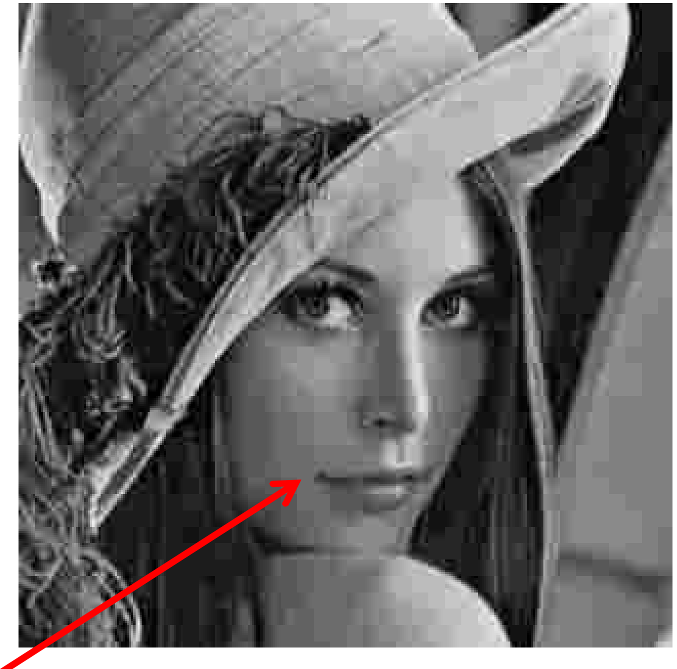
- Adjust Quantization Step to Achieve Tradeoff between CR and distortion.



Original: 100KB



JPEG: 9KB

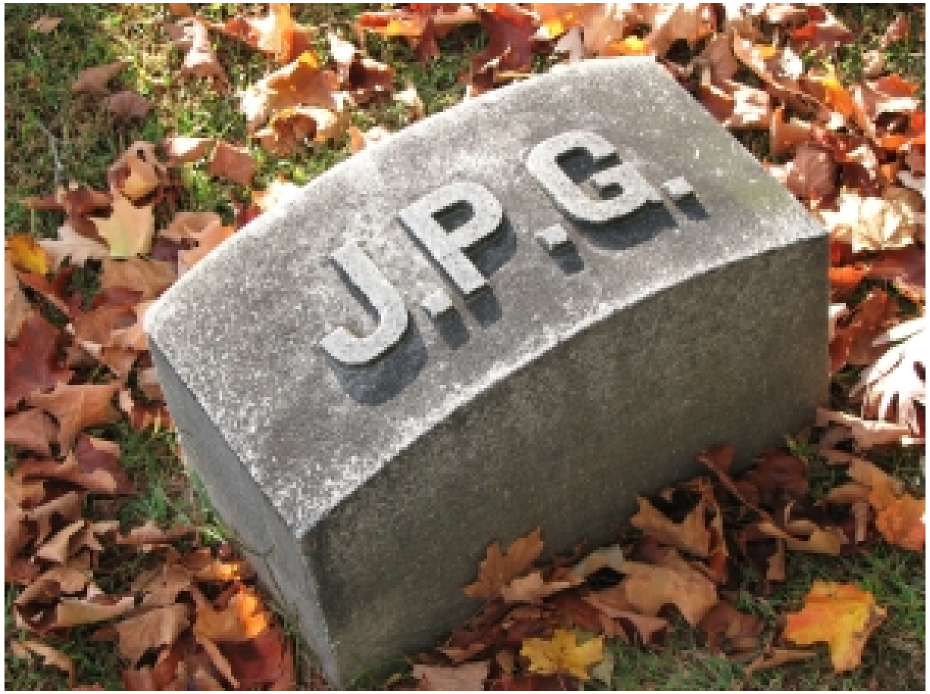


JPEG: 5KB

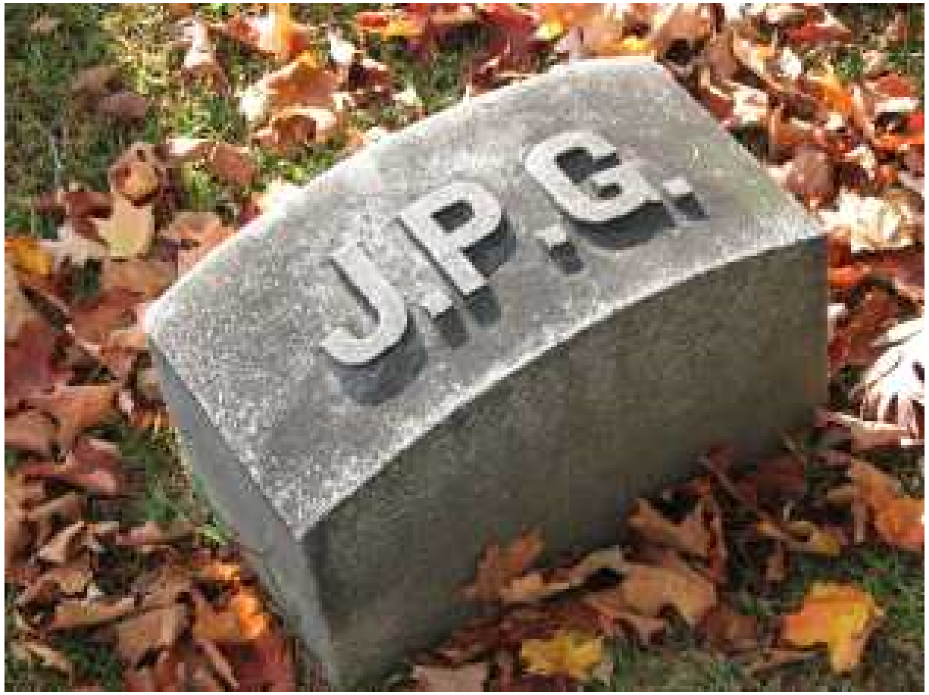
- Artifacts:

  Inside blocks: blurring (why?); Across blocks: blocking (why?)

JPEG compression results

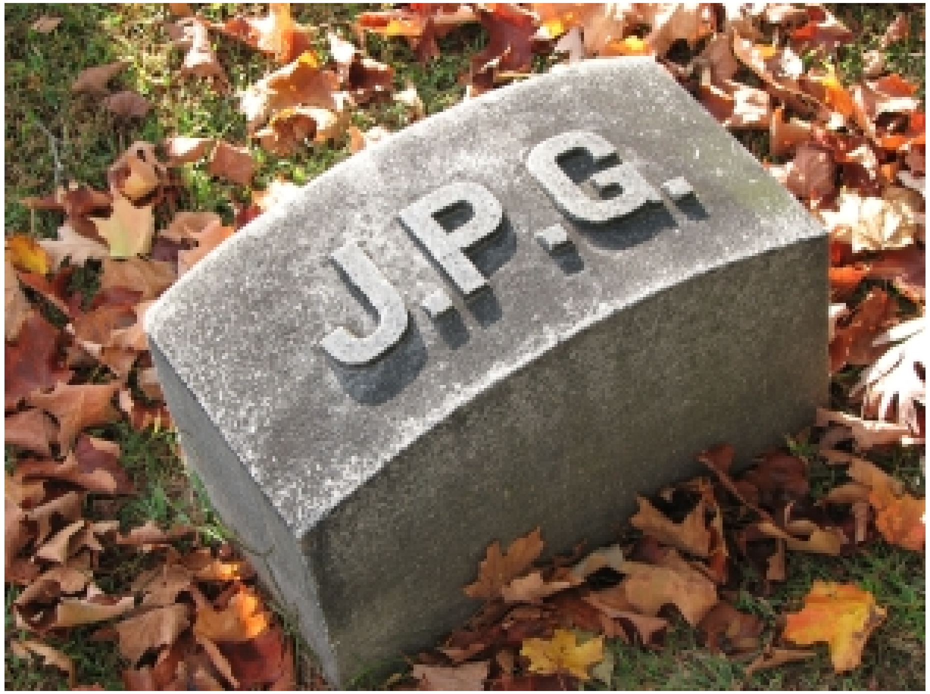


File size = 83,261 byte, Highest quality (Q = 100).

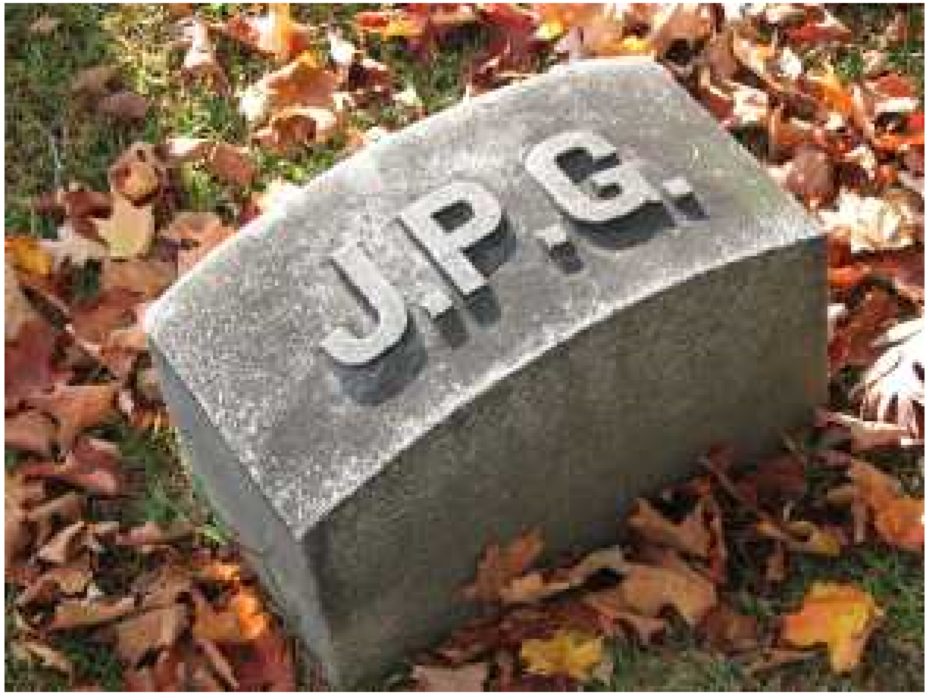


File size = 15,138 byte, High quality (Q = 50).

JPEG compression results

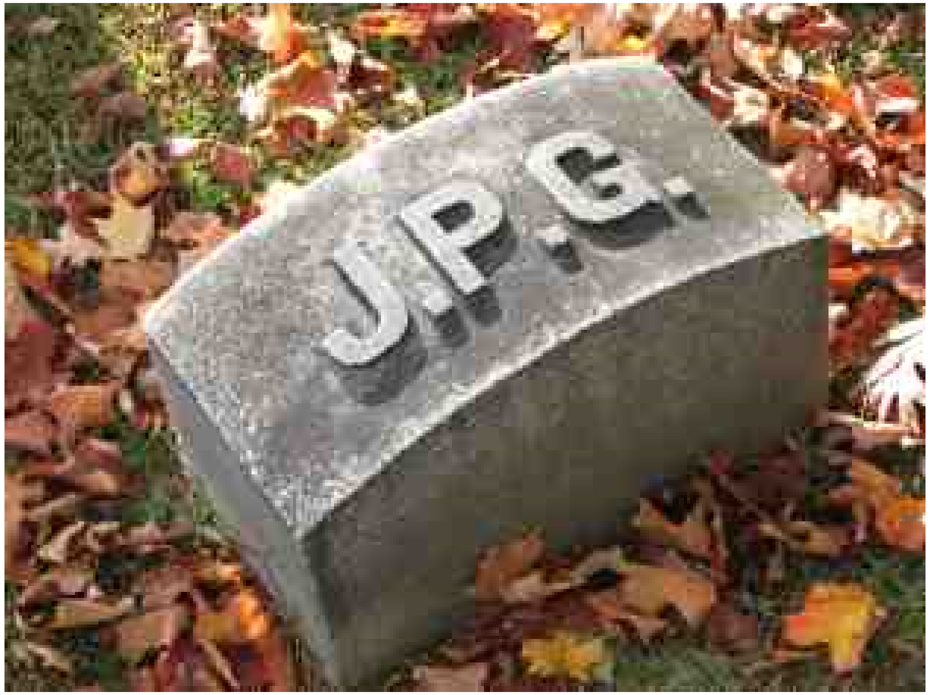


File size = 83,261 byte, Highest quality (Q = 100).

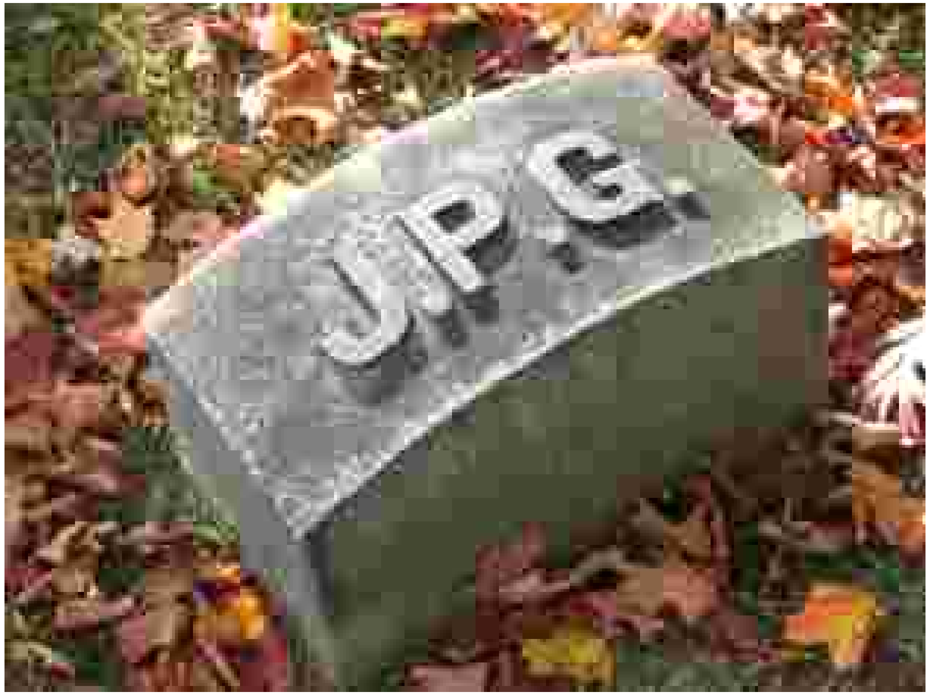


File size = 15,138 byte, High quality (Q = 50).

JPEG compression results

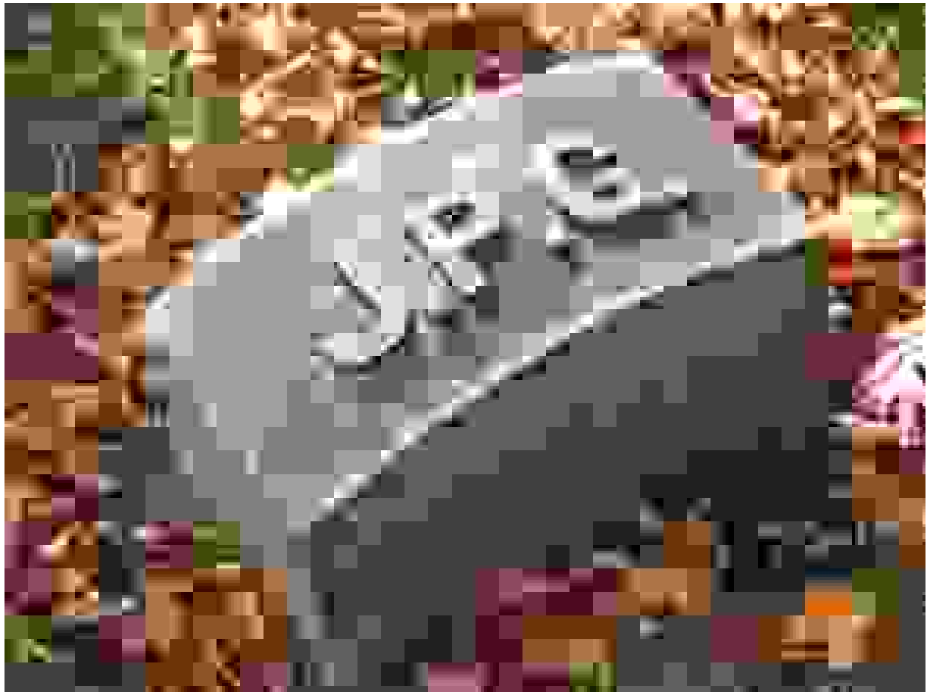


File size = 9,553 byte, Medium quality (Q = 25).



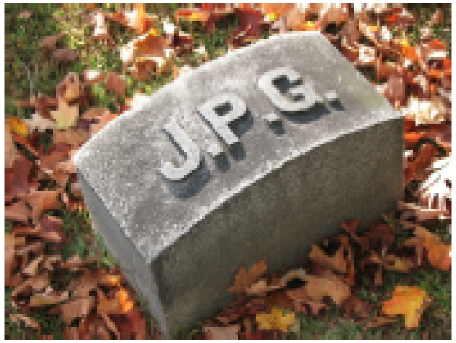
File size = 4,787 byte, Low quality (Q = 10).

JPEG compression results

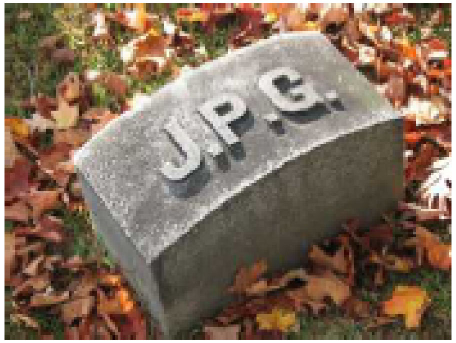


File size = 1,523 byte, Lowest quality (Q = 1).

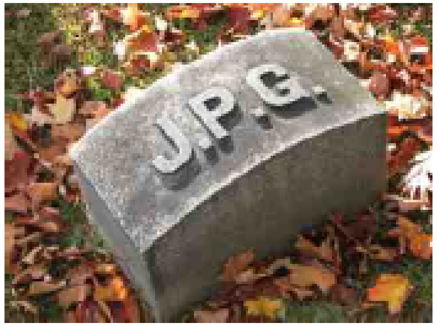
JPEG compression results



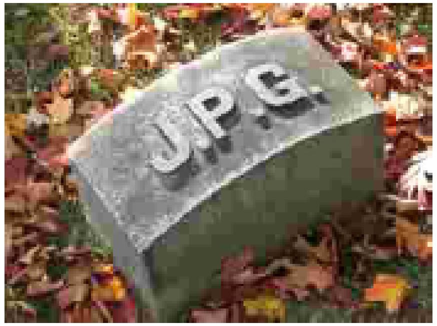
File size = 83,261 byte



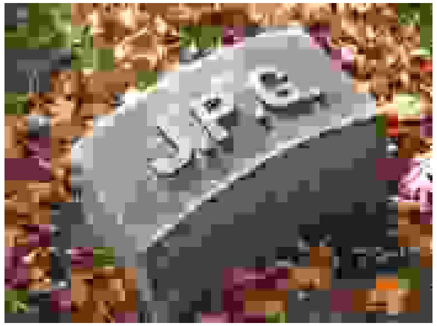
File size = 15,138 byte



File size = 9,553 byte



File size = 4,787 byte



File size = 1,523 byte

Thanks for your attention